Atlamalı Aralık Yayın Şifreleme Sisteminde Bedava Alıcıların İyi Yerleştirilmesi

Murat Ak[1] Ali Aydın Selçuk[2]

[1,2]Bilgisayar Mühendisliği Bölümü, Bilkent Üniversitesi, Ankara
[1]e-posta: muratak@cs.bilkent.edu.tr [2] e-posta: selcuk@cs.bilkent.edu.tr

Özetçe

Yayın şifreleme (YŞ) sistemlerinin amacı, çoklu bir alıcı kümesine yapılan yayınlarda sadece yayını satın almış olan alıcıların deşifre edebileceği şekilde bir şifreleme yapmaktır. Bu sistemdeki kaygılardan birisi yayının olabildiğince az sayıda kopyasının şifrelenmesidir çünkü her yeni şifreleme daha fazla bant genişliği gerektirmektedir. Alıcı kümesi kısıtlı miktarda bedava alıcıya izin verilmek suretiyle rahatlatılırsa şifreleme sayısı dikkate değer biçimde azaltılabilmektedir. Bu makalede literatürdeki en iyi performansa sahip YŞ metotlarından birisi olan Atlamalı Aralık metodu için bedava alıcıların optimal bir şekilde yerleştirilmesi problemi irdelenmiş, optimal yerleştirme algoritmasının yanısıra oldukça hızlı ve isabetli sonuç veren bir bulgusal metot da sunulmuştur.

1. Giriş

Yayın şifreleme (YŞ), büyük alıcı nüfusuna veri transferinde yayının sadece yayına erişim hakkı olan *imtiyazlı* bir alıcı kümesinin deşifre edebileceği şekilde şifrelenmesini amaçlayan kriptografik bir metoddur. Bu teknik genellikle ödemeli TV, içerik koruma, güvenli ses akışı, ve İnternet çoklu gönderimi gibi güvenlikli multimedya uygulamalarında kullanılmaktadır. Tipik olarak, YŞ sistemleri şöyle hayata geçirilir: Cihazlar kullanıcıya satılmadan önce birtakım anahtarlar içerisine gömülür. Yayın yapılacağında ise yayını deşifre edebilmesi gereken tüm kullanıcıların en az bir anahtarı bu yayını açabilecek, yayını açamaması gereken kullanıcılarınsa hiçbir anahtarı yayını açamayacak şekilde bir anahtar kümesi bulunarak şifrelemeler bu anahtarlarla yapılır. Değişik uygulamalarda alıcı küme büyüklüğü, donanım, bant genişliği gibi bazı kaygılar ön plana çıkabilir. Fakat, YŞ sistemlerinde en önemli kaygıların başında kullanıcıda tutulması gereken anahtar sayısı ve yapılması gereken şifreleme dolayısıyla yayınlanması gereken farklı şifreli yayın sayısı gelmektedir.

YŞ sistemlerinde iki ana değerlendirme kriteri vardır. Bunlardan birisi alıcı tarafında tutulması gereken anahtar sayısı, diğeri ise gönderi fazlalık maliyetidir. Günümüze dek önerilmiş en iyi performansa sahip YŞ sistemlerinden ilki Altküme Farkı (AF) metodu [21], ve türevleridir [11], [12]. AF metodu günümüzde de popüler hale gelmiş olup, yeni nesil DVD standardı gibi bazı uygulamalarda halen kullanılmaktadır. Daha sonra ise benzer parametrelere sahip başka YŞ sistemleri önerilmiştir. Bunlardan birisi de Jho vd. [16] tarafından önerilen Atlamalı Aralık (AA) metodudur.

Bir YŞ sisteminde, izinsiz kullanıcıların hiçbirinin yayını açamaması gerektiği genel bir kural olarak varsayılsa da, Abdalla vd. [2] bu varsayımın bazı uygulamalar için gereğinden fazla sıkı olduğunu, ve gönderi maliyetinin bu kural gevşetilerek ciddi ölçüde düşürülebileceğini farketmişlerdir.

1.1. İlgili Çalışmalar

YŞ problemi ilk defa Berkovitz [5] tarafından 1991 yılında ortaya atılmıştır. Daha sonra Fiat ve Naor'un modeli [10] ise YŞ'nin ilk formal modeli olarak kabul edilmiştir. Bu makalede ortaya dayanıklılık kavramını atmış, *k*-dayanıklılığı *k* tane izinsiz kullanıcının bir araya gelmesine karşın bilgi elde edememeleri şeklinde tanımlamışlardır. Önerdikleri en iyi metod, *n* toplam kullanıcı sayısı olmak üzere, her kullanıcının $O(k \log k \log n)$ anahtar tutmasını ve yayın merkezinin $O(k^2 \log^2 k \log n)$'lik mesajlar göndermesini gerektirmektedir.

1999 yılında, Wallner vd. [25] ve Wong vd. [26] birbirlerinden bağımsız biçimde Mantıksal Anahtar Hiyerarşisi'ni (MAH) önerdiler. MAH bir YŞ metodu değildi fakat kullandığı anahtar dağılım şekli YŞ sistemleri için çok uygundu. Bu fikre göre kullanıcılar bir ağacın yapraklarıyla, ağaç içerisindeki her düğüm ise bir anahtarla ilişkilendirilir. Daha sonra her kullanıcıya kök düğümden kendisine ulaşıncaya kadarki yoldaki düğümlerin anahtarları verilir. Bu yöntem Abdalla vd. [2] tarafından YŞ sistemleri için kullanılınca kullanıcılarda tutulan anahtar miktarı logaritmik mertebelere yani $O(\log n)$'e indirilmişti.

Kilometre taşı makalelerinde Naor vd. [21], iki metod önerdiler: Tam Altküme (TA) ve Altküme Farkı (AF) metodları. TA metodu aslında MAH fikrinin YŞ'ye uygunlanmasının güzel bir formalleştirilmesinden başka bir şey değildi. İzinsiz kullanıcı sayısı *r* olmak üzere, gönderi maliyeti $O(r \log(n/r))$ oluyordu. AF metodu ise $O(\log n)$ olan kullanıcı başına anahtar sayısını $O(\log^2 n)$'ye çıkarma karşılığında gönderi maliyeti $O(r)$'ye indirmeyi başarmıştı. Daha sonra AF'nin farklı türevleri önerildi. Önemli iki türevden birincisi olan Tabakalı AF (TAF), Halevy ve Shamir [12] tarafından önerildi. İdeal kullanımıyla, TAF $O(\log n \log\log n)$'lik bir gönderi maliyetini $O(r \log\log n)$'lik anahtar tutulmasıyla sağlıyordu. İkinci önemli türev olan Katmanlı AF (KAF) ise Goodrich vd. [11] tarafından önerildi. KAF ise $O(r \log n / \log\log n)$'lik gönderi maliyeti $O(\log n)$'lik anahtar tutumuyla sağlıyordu. Benzer sistemlerin [10],[12],[21] analizleri için [13]'e bakınız.

AF metodunun ve türevlerinin önerilmesinin ardından benzer performanslara sahip, Atlamalı Aralık (AA) gibi başka metotlar da önerildi. Jho vd. [16] tarafından önerilen AA metodu da altküme kaplama türünde bir YŞ sistemidir ve farklı bir altküme yapısına sahiptir. Bu sistemde altkümeler

tüm kullanıcılar bir doğru üzerinde hayal edilmek kaydıyla, belirli sayıda atlama (yani boş bırakılarak atlanmış kullanıcı pozisyonları) içerebilen belirli bir azami uzunluğa sahip aralıklar olarak tasarlanmıştır. Bununla birlikte bir şifrelemeyle daha fazla kullanıcıyı kaplayabilmek için tabakalı biçimde düşünülmüştür. Bunun da ötesinde, gönderi maliyetini bir miktar daha düşürebilmek için basamaklı bir türevi de önerilmiştir. Aslında bu son haliyle AA metodu AF metodunun bir genellemesi olarak ortaya çıkmaktadır.

Tüm bunlarla aynı zaman zarfında açık anahtarlı YŞ sistemleri de önerilmiştir. Bunlardan Boneh vd. [6] çiftdoğrusal fonksiyonlar ve çiftdoğrusal Diffie-Hellman üs kararı problemini kullanmıştır. Öte yandan, Boneh ve Hamburg [7] kimliğe dayalı YŞ sistemleri için zemin teşkil eden bir çalışma gerçekleştirdiler. Son zamanlarda, kimliğe dayalı ve açık-anahtar YŞ sistemlerine yoğun bir ilgi ile yaklaşıldı ve çeşitli çalışmalar yapıldı [8], [9], [17], [19], [22].

Aynı zamanda bir gevşetme olarak da görülebilecek olan bedavacı fikri ilk defa Abdalla vd. [2] tarafından önerildi. Bu makalede, bedavacıları etkili biçimde kullanma problemi derinlemesine irdelendi ve bu konunun temelleri atılmış oldu. Ramzan ve Woodruff [23], yakın zamanda TA metodu için bedavacıların kullanımı problemine bir eniyileme algoritması verdiler. Algoritmaları, dinamik programlamaya dayanıyordu. Daha yakın bir zamanda ise Ak vd. [4] aynı problemin AF metodu için olanını çözdüler.

1.2. Katkılar

Bu makalede, AA metodu için bedava alıcıları en etkili biçimde seçerek gönderi maliyetini olabildiğince azaltma problemini ele alıyoruz. İlk olarak, bu metotta, bedavacıların en iyi nasıl yerleştirilebileceğini bulan optimal bir algoritma veriyoruz. Daha sonra ise çok daha hızlı çalışmasına rağmen optimale yakın sonuç veren Yukarıdan Aşağıya (YA) adında bulgusal bir prosedür ortaya koyuyoruz. Son olarak da deneysel sonuçları sunuyoruz.

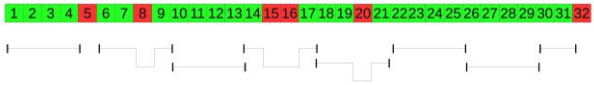
1.3. Yerleşim

Öncelikle 2. bölümde AA metodunu kullanılan şekliyle tanıtıyoruz. 3. bölümde çalışılan problemi formal biçimde ortaya koyuyoruz. 4. bölümde eniyileme algoritmasını, 5. bölümde bulgusal prosedürümüzü açıklıyoruz. 6. bölümde deneysel sonuçları sunduktan sonra 7. bölümdeyse sonuçları yorumlayarak makaleyi bitiriyoruz.

2. Atlamalı Aralık (AA) Metodu

AA metodunda altkümeler, içlerinde *sınırlı sayıda* boşluk bulunan *sınırlı uzunlukta* aralıklar olarak tasarlanmıştır. İşte AA metodundaki başlıca iki parametre olan *a* ve *b* sırasıyla bu azami aralık uzunluğu ve azami boşluk sayısını temsil etmektedir. Dolayısıyla, kullanıcılar hayali bir doğru üzerinde düşünüldüğünde, söz konusu altkümeler, bu parametrelerle oluşacak olan atlamalı aralıklarda bulunan kullanıcıların oluşturduğu altkümelerdir. Bu altkümeler, $S_{i,j;B}$ ile gösterilecektir. Bu gösterimde *i* aralığın başlangıcını, *j* ise sonunu temsil eden sayılar, *B* ise bunlar arasındaki boşlukları ifade eden azami *b* büyüklüğünde bir sayı kümesidir. Örneğin, $S_{3,9;5,8}$ altkümesi denince 3, 4, 6, 7, 9 numaralı kullanıcılar anlaşılmalıdır (Şekil 1). Örnek bir kaplama kümesi ise Şekil 2'de gösterilmiştir.

Şekil 1:Örnek AA altkümesi.



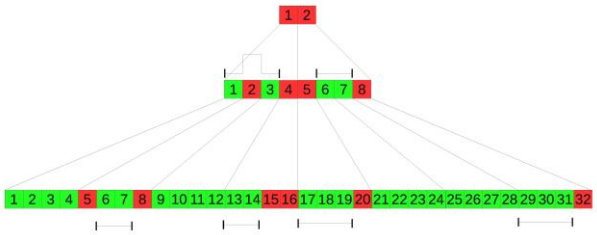
Şekil 2: Örnek AA kaplama kümesi.

Bu şekilde yazılabilecek olan altküme sayısı oldukça çok olsa da orijinal makalede tek yönlü fonksiyonlar kullanılarak kullanıcıda tutulması gereken anahtar sayısı düşürülmüştür. Bu makalede incelenen problem tutulan anahtar sayılarından bağımsız olduğu için bu konuda detaya inmeyi gerekli bulmuyoruz.

Altkümeler bu şekilde tasarlandıktan ve anahtarlar bunlara göre dağıtıldıktan sonra, yayın yapılacağında tek yapılması gereken, imtiyazlı küme için bir kaplama bileşimi bulmaktır. Bu ise en baştan başlayarak tam olarak imtiyazlı kullanıcıları alacak şekilde teker teker gerekli altkümeleri alarak yapılabilecek basit bir işlemdir.

2.1. Katmanlı AA Metodu

Dikkat edilirse, kullanıcıların çoğunun imtiyazlı olması durumunda yukarıda anlatılan temel yöntemde kaplama birçok yanyana atlamasız altküme içerecektir. Dolayısıyla [16] metoda katmanlar ekleyerek bu sorunu çözme yoluna gitmiştir. Bu metotta temel fikir şöyledir: Her katmandaki *a* tane kullanıcı, bir üst katmanda tek kullanıcı olarak düşünülür. Böylece her biri bir üstündekinin *a* katı sayıda kullanıcı içeren $\lfloor \log_a n \rfloor + 1$ tane katman oluşur. Üst katmandaki aralıklar da aynen sıfırıncı katmandaki gibi düşünülür. Örnek bir kaplama Şekil 3'de verilmiştir.



Şekil 3: Örnek katmanlı AA kaplama kümesi.

Katmanlı AA metodunda aralıklar bulundukları katmanı ifade eden yeni bir parametreye daha sahiptirler. $S_{\ell;i,j;B}$ şeklinde ifade edilen bir aralık, ℓ'inci katmandaki $S_{i,j;B}$ aralığını ifade eder.

Kolayca görülebilir ki, katmanlı AA metodunda en iyi kaplamayı bulmak için en üst katmandan başlayarak aşağıya doğru tek katmanlıda olduğu gibi kaplamalar yapılarak

inilmesi, bir katman bitince alt katmanlarca kaplanması gerekmeyeceğinden kaplanmış olanların kaplanmış olarak işaretlenmesi yeterlidir. Bunun en küçük kaplama olduğu aşikardır çünkü bir katmandaki en uzun aralık bir üst katmanda sadece bir düğüme denk gelmektedir.

Jho vd. [16] bunların üzerine basamaklama yöntemini de kullanmışlardır ki böylelikle farklı katmanlardaki aralıkların birbirine eklenmesiyle elde edilen aralıklar da altküme yapısına dahil olmuştur. Fakat bu, ancak % 0.1'den az izinsiz bulunduğunda faydalı olduğundan makalenin olabildiğince açık olması için ele alınmamış, katmanlı AA metodu üzerinde yoğunlaşılmıştır.

3. Problem Tanımı

Tüm altküme kaplama metotlarında olduğu gibi, AA metodunda da belirli sayıda bedavacıya izin verilmek suretiyle gönderi maliyetini düşürmek mümkündür. Burada ortaya çıkan soru şudur: Belirli bir oranda bedavacıya izin verildiği takdirde gönderi maliyeti olabildiğince etkili ve verimli bir biçimde nasıl düşürülebilir?

Problem tanımında ve makalenin geri kalanında kullanacağımız gösterimler şu şekildedir: Tüm kullanıcıların kümesini *U*, izinsizlerin kümesini *R*, imtiyazlıların kümesiniyse *P* ile göstereceğiz. Ayrıca $n = |U|$, $r = |R|$, $p = |P|$ olarak tanımlanacak. Bedavacıların izinsizlere oranını c_f ile gösteriyoruz. İzin verilecek olan toplam bedavacı sayısını ise *f* ile göstereceğiz. Dolayısıyla $f = r \cdot c_f$ olacak. Bölüm 2.1'de anlatıldığı şekilde, tüm kaplayıcı altkümelerin kümesine *S* diyeceğiz. Hatırlanacağı üzere katmanlı AA metodunda altkümeler $S_{\ell,i,j,B}$ şeklinde gösteriliyordu. Değişkenleri bu şekilde isimlendirdikten sonra problemi ise şöyle tanımlıyoruz:

$$P \subseteq \bigcup_{S_{\ell;i,j;B} \in C} S_{\ell;i,j;B} \quad \text{ve} \quad \left| \bigcup_{S_{\ell;i,j;B} \in C} S_{\ell;i,j;B} \setminus P \right| \leq f = c_f \cdot r$$

olacak şekilde en küçük $C \subseteq S$'yi bulunuz. Bir başka deyişle, imtiyazlı kullanıcıları tamamen kapsayacak, fakat izinsiz kullanıcılardan en fazla *f* tanesini içerecek en küçük kaplamayı bulunuz.

4. Optimal Yöntem

Bu bölümde yukarıdaki problem için dinamik programlama ile eniyileme algoritmasını sunacağız.

4.1. Gösterimler

Algoritmaya geçmeden önce kullanacağımız gösterimleri ortaya koyacağız.

Algoritmanın üzerinde çalışacağı temel veriyapısı *KA* (kullanıcı ağacı) ismiyle iki boyutlu bir matris olacaktır. *KA*[*k*], *k*'ıncı düğüm katmanı olan KA'nın *k*'ıncı satırını, *KA*[*k*][*i*] ise *KA*[*k*] katmanındaki *i*'inci düğümü gösterecektir. *KA* veriyapısı aslında Şekil 3'deki yapıyı ifade eder. Aynı zamanda her düğümün bir alt katmandaki sol ve sağ çocuklarına işaretçileri algoritmada ".sol" ve ".sağ" şeklinde eklentilerle gösterilecektir. Ayrıca, ".im" ve ".iz" de düğüm altındaki sırasıyla imtiyazlı ve izinsiz kullanıcı sayısını ifade edecektir. Örneğin, Şekil 3'ye göre, *KA*[1][4].sol = 13, *KA*[1][4].im = 2 olmalıdır.

Dinamik programlamanın dolduracağı veriyapısı ise dört boyutlu *maliyet* matrisi olacaktır. *maliyet*[*k*][*s*][*e*][*f*], *k*'ıncı katmandaki (*s*,*e*) aralığını en çok *f* adet bedavacı ile kaplamanın minimum maliyeti olsun. Algoritma, $0 \leq k \leq \lfloor \log_a n \rfloor + 1$, $0 \leq i \leq j \leq |KA[k]|$ ve $0 \leq fr \leq f$ değerleri için *maliyet* tablosunu dolduracaktır.

Bunların yanısıra, algoritmada azami *a* uzunluğunda en çok *b* atlama içeren olası tüm atlamalı aralık kombinasyonları *AA* ile gösterilecektir. *AA* içerisindeki herhangi bir eleman için *A* değişkeni kullanılıp, *A.atla* ile *A*'nın içindeki atlama aralıklarının (atlanılan düğüm veya düğüm dizilerinin) kümesi kastedilecektir. Mesela A = $S_{3,9;5,8}$ aralığı için *A.atla* = {[5,5],[8,8]} olacaktır. Fakat A = $S_{3,9;5,7,8}$ olsaydı *A.atla* = {[5,5],[7,8]} olacaktı. Bu aralıkların başları ve sonları ise *.baş* ve *.son* eklentileri ile gösterilecektir. Örneğin *at* = [7,8] ise *at.baş* 7, *at.son* 8 olacaktır.

4.2. Algoritma

Bu bölümde, izinsiz kullanıcılardan belli bir bedavacı oranına izin vererek gönderi maliyetini en çok düşüren algoritmayı vereceğiz. Sözkonusu eniyi kaplama algoritması Algoritma 1'de, yardımcı prosedürü `MaliyetBul` ise Algoritma 2'de verilmiştir.

Algoritma 1. `Eniyileme`
Girdiler: *c, a, U, P,R,* c_f
1: $k_a \leftarrow \lfloor \log_a n \rfloor + 1$
2: *KA* ← `KullanıcıAğacıOluştur`
3: *AA* ← `AtlamalıAralıkKombinasyonlarınıYaz`
4: $f \leftarrow c_f$. $|R|$
5: **eğer** $f \geq |R|$ **ise**
6: **döndür** 1
7: *k* ← 0**'dan** k_a**'ya kadar**
8: *son* ← |*KA*[*k*]|**'dan** 0**'a kadar**
9: *baş* ← *son***'dan** 0**'a kadar**
10: *fr* ← 0**'dan** *f***'ye kadar**
11: *maliyet*[*k*][*baş*][*son*][*fr*] ← `MaliyetBul`(*k*,*baş*,*son*,*fr*)
12: **döndür** *maliyet*[k_a][0][0][*f*]

Eniyi kaplama algoritması şöyle çalışmaktadır: Öncelikle `KullanıcıAğacıOluştur` prosedürü kullanıcıları katmanlar halinde bir veriyapısı (*KA*) şeklinde düzenler.

Daha sonra kaç tane bedavacıya izin verileceği, verilen c_f oranından hesaplanır. Öncelikle bir kısayol olarak, eğer bedavacı sayısı izinsizlere eşit olabiliyorsa, bir başka deyişle sözkonusu oran 1 ise, herkesi birden en üst düğüm ile alarak yanıt kaplama büyüklüğü olarak 1 döndürülür.

Aksi halde tablo şu şekilde doldurulur:
En alt katmandan başlanıp en üst katmana doğru ilerlenir. Her katmanda olabilecek her ikili arası için belli sayıda

bedavacıya izin verildiği takdirde elde edilecek kaplama büyüklükleri hesaplanacak şekilde döngüler hazırlanır ve sözkonusu her aralık ve bedavacı sayısı için **`MaliyetBul`** prosedürü çalıştırılır. Tablo dolduğunda, yanıt en üst katmanın ilk ve tek düğümünün tablodaki maliyetidir.

`MaliyetBul` Algoritmasını vermeden önce incelenen aralığın maliyetini bulurken belli bir başlangıç düğümü *s* ve belli bir atlamalı aralık *A* için yapılan hesaplamayı *MinMlyt* fonksiyonu ile şöyle buluyoruz:

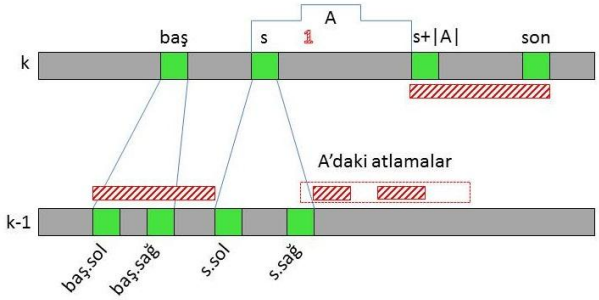
$$MinMlyt(s,A) = \min_{fsol+\,fsağ+\,falt=\,fr} \left\{ \begin{array}{l} 1+ maliyet(k-1, baş.sol, s.sol-1,\ fsol) \\ + maliyet(k, s+|A|, son,\ fsağ) \\ + \min_{\substack{falt=\,fr-\,fsol-\,fsağ \\ \sum_{at\in A.atla} f_{at}=\,falt}} \left\{ \sum_{at\in A.atla} maliyet(k-1, at.baş.sol, at.son.sağ,\ f_{at}) \right\} \end{array} \right\}$$

MinMlyt fonksiyonu, aşağıdaki **`MaliyetBul`** algoritmasında, temel bir yapıtaşı olarak kullanılacaktır.

Algoritma 2. **`MaliyetBul`**
Girdiler: *k, baş, son, fr*
1: **eğer** $\sum_{i=baş}^{son} KA[k][i].iz < fr$ **ise**
2: **döndür** sonsuz
3: **eğer** $\sum_{i=baş}^{son} KA[k][i].im = 0$ **ise**
4: **döndür** 0
5: **eğer** baş = son **ise**
6: **eğer** *fr* = *KA*[*k*][*baş*].iz **ise**
7: **döndür** 1
8: **eğer** *fr* < *KA*[*k*][*baş*].iz **ise**
9: **eğer** *k* > 0 **ise**
10: çsol ← KA[k][*baş*].sol
11: çsağ ← KA[k][*son*].sağ
12: **döndür** *maliyet*[*k*–1][çsol][çsağ][*fr*]
13: **döndür** $\min_{baş\le s\le son; A\in AA}\{MinMlyt(s,A)\}$

`MaliyetBul` prosedürü girdilerinden anlaşılacağı üzere belli bir katmandaki belli iki düğüm arasına belli sayıda bedavacı verilmesi durumunda burayı kaplamak için gerekecek maliyeti hesaplayıp tabloya yazmakla görevlidir. Bu prosedür iki kısma ayrılabilir: Birincisi temel durumları inceleyen ilk 12 satır, ikincisi ise tablodan yararlanarak özyinelemeli çağrıyı yapan 13. satırdır.

Temel durumlarda, basit kurallar ile hesaplanabilen temel maliyetler tabloya işlenir. Örneğin bir aralığa buradaki izinsiz kullanıcılardan daha fazla bedavacı verilmişse tabloya sonsuz maliyet işlenir. Bir aralıkta hiç imtiyazlı yok ise ve verilen bedavacı sayısı 0 ise tabloya 0 maliyet işlenir. Veya tek düğümlük bir aralık ise verilen bedavacı sayısının yetip yetmemesine göre 1 maliyet işlenir veya alt katmanın tablo kaydından maliyet aktarılır.



Şekil 4: MinMlyt hesabının gösterimi. Taralı alanların her biri MinMlyt hesabındaki toplanan terimlerden birisine karşılık gelmektedir.

Eğer durum temel bir durum değilse, daha önce hesaplanmış tablo değerlerinden minimum maliyet hesaplanır. Burada yapılan, tüm maliyetlerin daha önce hesaplanıp tabloya yazılmış olmasından yararlanarak bu maliyetler arasında en düşük maliyeti bularak tabloya yazmaktır.Şekil 4'te belli bir başlangıç noktası *s* ve aralık *A* için minimum maliyeti bulan *MinMlyt* işlemi görsel olarak açıklanmaktadır. Algoritmada sadelik için *KA*[*k*-1][*baş*].sol yerine *baş*.sol, *KA*[*k*–1][*s*].sol yerine *s*.sol, *KA*[*k*][*s*+|*A*|] yerine *s*+|*A*|, *KA*[*k*][*son*] yerine *son* gibi kullanımlar yapılmıştır.

Sonuç bulunurken tüm $0 < f r < f$ değerlerini ele almaya gerek kalmamaktadır çünkü *maliyet*[*k*][*s*][*e*][*f*], en çok *f* adet bedavacıya izin verilen durumdaki minimum maliyeti ifade ettiği için, *maliyet*[*k*][*s*][*e*][*f*] değerlerinin *f* ile artmayan olması garantilenmiş olmaktadır. Böylece sadece *f* için ortaya çıkan sonuç yeterlidir.

Dikkat edileceği üzere yazmış olduğumuz eniyileme algoritması kaplamanın büyüklüğünü bulmaktadır. Fakat bedavacıların yerleşimleri de tabloda gerekli iz sürme bilgileri tutularak kolayca bulunabilir.

4.3. Performans üzerine

Eniyileme algoritması en düşük maliyetli bedavacı yerleşimini bulmasına karşın, çalışma hızı düşük olmaktadır. Doldurulan dinamik programlama tablosunun boyutları $\log(n)\times n\times n\times f$ dir. Her ne kadar tablonun tümü doldurulmasa da kuramsal olarak $O(\log(n)\ n^2\ f)$ defa MaliyetBul prosedürü çağrılmaktadır. Bu prosedürün çalışma hızı da $O(n \binom{a}{b})$ olacağı için toplam zaman maliyeti $O(\log(n)\ n^3\ f \binom{a}{b})$ olacaktır.

Zaman maliyetini azaltmak için çeşitli stratejiler uygulanabilir. Bunlardan birisi şudur: Herhangi bir katmanda tamamen imtiyazlı kullanıcılar içeren bir düğüm var ise, en iyi çözümlerden birinin bu düğümü bu katmanda kaplaması gerektiği aşikardır. Bu durumda bir alt katmanda bu düğümün solundan başlayıp sağında biten aralıkların hesaplanması

gereksizdir. Dolayısıyla bu aralıklara ilişkin maliyetlerin gerek hesaplanması, gerek tabloya yazılması gerekse maliyet hesap ederken bakılmasına gerek kalmamaktadır. Ne var ki bu sadece ortalama karmaşıklığı azaltmakla beraber en kötü durum karmaşıklığı aynı kalmaktadır.

5. Yukarıdan Aşağıya: Açgözlü bulgusal yöntem

Bu bölümde eniyileme algoritmasına göre çok daha hızlı çalışırken gönderi maliyetini da oldukça iyi ölçüde düşüren bulgusal bir yöntem sunuyoruz. Bu bulgusal yöntem temel olarak şöyle çalışır: En üst katmandan başlamak suretiyle en alt katmana kadar `KatmanKapla` prosedürüyle her katmanı kaplayarak ilerler. (Bkz. Algoritma 3.) Herhangi bir katman içinse `KatmanKapla` prosedürünün işleyişi Algoritma 4'te verilmiştir.

Algoritma 3. `YukarıdanAşağı`
Girdiler: c_f
1: $k_a \leftarrow \lfloor \log_a n \rfloor + 1$
2: $KA \leftarrow$ `KullanıcıAğacıOluştur`
3: $AA \leftarrow$ `AtlamalıAralıkKombinasyonlarınıYaz`
4: $f \leftarrow c_f \,.\, r$
5: $d_f \leftarrow f / p$
6: $C \leftarrow \{\}$
7: $k \leftarrow k_a$'**dan** 0'**a kadar**
8: `KatmanKapla(`k,d_f`)`
9: **döndür** $|C|$

`YukarıdanAşağı` prosedürü öncelikle kaç bedavacıya izin verilebileceğini bulur. Daha sonra bu sayıyı toplam imtiyazlı sayısına bölerek kaç imtiyazlı kullanıcıyı kaplamak için kaç bedavacıya izin verilebileceğine dair bir oran bulur. Bu yeni orana d_f diyoruz. Daha sonra en üst katmandan en alt katmana kadar her katmanı bu orana göre ele alması için `KatmanKapla` prosedürünü çağırır.

Algoritma 4. `KatmanKapla`
Girdiler: k, d_f
1: $sıra \leftarrow 0$
2: $sıra \leq |KA[k]|$ **olduğu sürece**
3: **sırayla tüm** *sıra* ile başlayan $A \in AA$ **için**:
4: **eğer** $izinsiz(A)/imtiyazlı(A) \leq d_f$ **ise**
5: $C \leftarrow C \cup A$
6: `KaplandıİşaretleA`
7: $sıra \leftarrow sıra + |A| - 1$
8: **döngüyüDurdur**
9: $sıra \leftarrow sıra + 1$

`KatmanKapla` prosedürü belirli bir katman için şöyle çalışır: En baştan başlayarak belli bir *sıra* düğümünden başlayan olabilecek atlamaları aralıklara birer birer bakar ve d_f oranını sağlayan bir atlamalı aralık bulunur bulunmaz bu aralığı kaplamaya ekler ve bu aralığın bir ardındaki düğümden devam eder. Eğer hiçbir atlamalı aralık d_f'yi sağlamazsa *sıra*'yı bir sağa kaydırır; ta ki bu katmandaki tüm düğümler bitene kadar. Bu prosedürde atlamalı aralıklar alındığı takdirde kaplanacak olan izinsiz kullanıcı sayısı $izinsiz(A)$, imtiyazlı kullanıcı sayısı ise $imtiyazlı(A)$ notasyonuyla gösterilmiştir. Benzer şekilde, $izinsiz(C)$ de halihazırda kaplanmış küme olan C'deki izinsiz kullanıcı sayısını, bir başka deyişle halihazırdaki bedavacı sayısını ifade etmektedir.

Bu yöntemle elde edilen sonucun c_f oranının izin verdiği ölçüde bedavacıyı aşmasının mümkün olamayacağı ise d_f'in hesaplanma şeklinden dolayı kesindir. Çünkü öncelikle cf oranını sağlayacak bedavacı sayısı, f, bulunmakta, d_f ise bu sayının imtiyazlılara bölümüyle elde edilmektedir. Yöntemde kaplanan tüm A aralıkları bağımsız olarak d_f oranını sağladıkları için bu aralıkların tamamının toplamı ele alındığında aynı oranın korunacağı ortadadır.

5.1. Performans üzerine

`YukarıdanAşağı` bulgusal metodu, oldukça hızlı ve akla yatkın olmasına karşın, küçük bir sorun vardır. İzin verilen bedavacı sayısının az olması katmanlı yapının sağlayabileceği faydayı sınırlı kılmaktadır.

Bu sorunun üstesinden gelmek için *tolerans* adında yeni bir parametre tanımlıyoruz ve Algoritma 4, 5. satırdaki açgözlü seçimi yaparken sadece d_f yerine $tolerans \cdot d_f$ çarpımını kullanıyoruz. Elbette bununla beraber aralığın alınması durumunda toplam bedavacı sayısının geçilip geçilmeyeceği de kontrol ediliyor ve eğer geçecekse bu aralık alınmıyor. Dolayısıyla, Algoritma 4'ün 4. Satırı şu şekilde değişiyor:

4: **eğer** $izinsiz(A)/imtiyazlı(A) \leq tolerans \cdot d_f$ ve $izinsiz(A) + izinsiz(C) \leq f$ **ise**

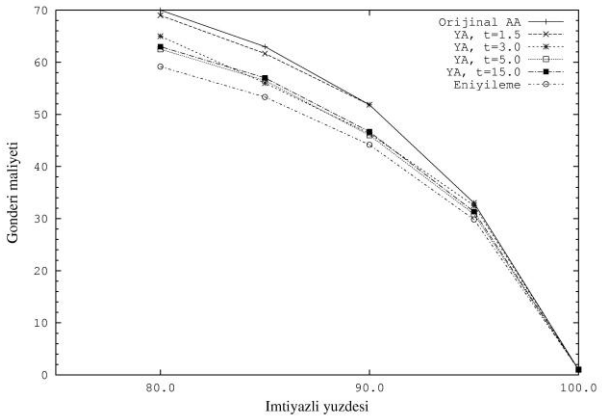
Bu değişikliğin gönderim maliyeti düşürmedeki etkisi Deneyler bölümünde görülebilir.

`YukarıdanAşağı` bulgusal metodunda $\log(n)$ defa `KatmanKapla` prosedürü çalıştırıldığı ve bu prosedürde sırayla mümkün olan tüm atlamalı aralıklar kontrol edildiği için, en kötü durumda çalışma hızı $O(\log(n)\, n \binom{a}{b})$ olmaktadır.

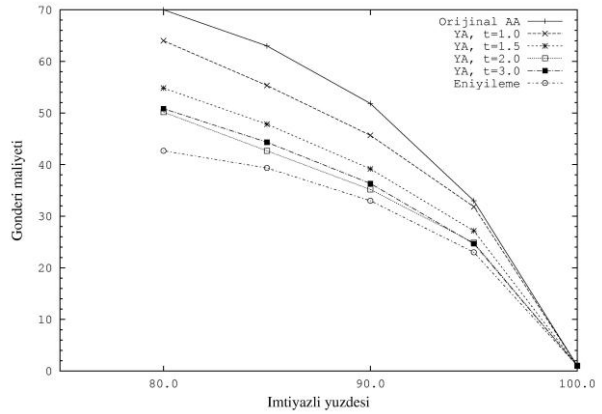
6. Deneyler

Bu bölümde önerilen çözümlerin deneysel değerlendirmesi sunulacaktır. Şekil 5,6,7,8'deki grafiklerde, %70 ila %100 arası imtiyazlı kullanıcı içeren 1024 kullanıcılık popülasyonlarda a=4,8,16; b=1,2 durumları gösterilmektedir. Grafikler, imtiyazlı kullanıcı yüzdesinin fonksiyonu olarak gönderi maliyetlerini, normal AA yöntemi ile karşılaştırmaktadır. Şekiller incelendiğinde görülebilir ki eniyileme algoritması izin verilen bedavacı oranına bağlı olarak, gönderi maliyetinde dikkate değer bir azalma sağlamaktadır. Şekil 8'de görülebilir ki 0.3'lük bir bedavacı oranıyla gönderi maliyeti yarıdan aza inebilmektedir.

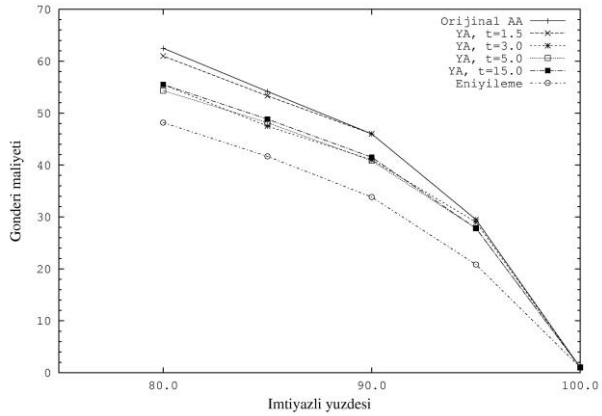
Öte yandan, çok daha pratik olan yukarıdan aşağıya açgözlü bulgusal yöntemi ile eniyileme algoritmasına oldukça yakın sonuçlar elde edilebildiği de gözden kaçmamaktadır. Grafiklerin hemen hepsinde bulgusal yöntemin optimalin sağladığı avantajın yarıdan fazlasını sağlayabildiği görülmektedir.



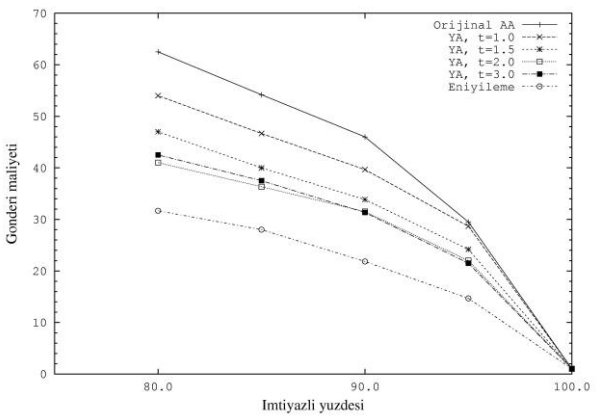
Şekil 5: a=8, b=1, c_f=0.1 için orijinal AA, farklı tolerans değerleri için **YukarıdanAşağı** *metodu, ve eniyileme algoritmasının sonuçları.*



Şekil 6: a=8, b=1, c_f=0.3 için sonuçlar.



Şekil 7: a=8, b=2, c_f=0.1 için sonuçlar.



Şekil 8: a=8, b=2, c_f=0.3 için sonuçlar.

7. Sonuç

Bu makalede AA metodunda bedavacıların, gönderi maliyetini düşürmek için en etkili bir biçimde nasıl kullanılabileceği incelenmiştir. Eniyileme algoritmasının yanısıra parametrelendirilmiş bulgusal bir prosedür olan **YukarıdanAşağı** da önerilmiştir. Eniyileme algoritmasının kuramsal zaman karmaşıklığı $O(\log(n)\ n^3\ f\binom{a}{b})$, bulgusal **YukarıdanAşağı**'nınki ise $O(\log(n)\ n\ \binom{a}{b})$'dir. Bu yöntemlerin gönderi maliyetinde sağladıkları düşüş ise deneylerle ortaya konmuştur. Halen **YukarıdanAşağı** metodu ile aynı mertebede hıza sahip olup en iyi sonuca daha yakın bir yöntem bulma problemi açıktır.

8. Kaynakça

[1] AACS - Advanced Access Content System, http://www.aacsla.com, 2007.

[2] M. Abdalla, Y. Shavitt, and A. Wool. Key management for restricted multicast using broadcast encryption. *IEEE/ACM Trans. Networking*, 8(4):443--454, 2000.

[3] W. Aiello, S. Lodha, and R. Ostrovsky. Fast digital identity revocation. In *CRYPTO'98*, volume 1462 of *LNCS*, pages 137--152. Springer-Verlag, 1998.

[4] M. Ak, K. Kaya, A. A. Selcuk. Optimal Subset-Difference Broadcast Encryption with Free Riders. *Information Sciences* Volume: 179, Issue: 20, Publisher: Elsevier Inc., Pages: 3673--3684

[5] S. Berkovits. How to broadcast a secret. In *EUROCRYPT'91*, volume 547 of *LNCS*, pages 535--541. Springer-Verlag, 1991.

[6] D. Boneh, C. Gentry, and B. Waters. Collusion resistent broadcast encryption with shorter ciphertexts and private keys. In *CRYPTO'05*,

volume 3621 of *LNCS*, pages 258--275. Springer-Verlag, 2005.

[7] D. Boneh and M. Hamburg. Generalized identity based and broadcast encryption schemes. In *ASIACRYPT'08*, volume 5350 of *LNCS*, pages 455--470. Springer-Berlin, 2008.

[8] V. Daza, J. Herranz, P. Morillo, and C. Ráfols. Ad-hoc threshold broadcast encryption with shorter ciphertexts. *Electronic Notes in Theoretical Computer Science*, 192(2):3--15, 2008.

[9] C. Delerableé and D. Pointcheval. Dynamic threshold public key broadcast encryption. In *CRYPTO'08*, volume 5157 of *LNCS*, pages 317--334. Springer-Verlag, 2008.

[10] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO'93*, volume 773 of *LNCS*, pages 480--491. Springer-Verlag, 1993.

[11] M. T. Goodrich, J. Z. Sun, and R. Tamassia. Efficient tree based revocation in groups of low-state devices. In *CRYPTO'04*, volume 3152 of *LNCS*, pages 511--527. Springer-Verlag, 2004.

[12] D. Halevy and A. Shamir. The LSD broadcast encryption scheme. In *CRYPTO'02*, volume 2442 of *LNCS*, pages 47--60, London, UK, 2002. Springer-Verlag.

[13] J. Horwitz. A survey of broadcast encryption, 2003. Manuscript.

[14] N. Jho, E. Yoo, J. Cheon, and M. Kim. New broadcast encryption scheme using tree-based circle. in Proc. of ACM Digital Rights Management Workshop, 2005, pp.37-44.

[15] N. Jho, J. Hwang, J. Cheon, M. Kim, D. Lee, and E. Yoo, "One-way Chain Based Broadcast Encryption Schemes," Proc. of EUROCRYPT 2005, LNCS 3494, pp. 559-574, 2005

[16] J. Cheon, N. Jho, M. Kim, and E. Yoo, "Skipping, Cascade, and Combined Chain Schemes for Broadcast Encryption," IEEE Transaction on Information Theory 54(11), pp. 5155-5171, 2008

[17] M. Kusakawa, H. Hiwatari, T. Asano, and S. Matsuda. Efficient dynamic broadcast encryption and its extension to authenticated dynamic broadcast encryption. In *CANS'08*, volume 5339 of *LNCS*, pages 31--48. Springer-Verlag, 2008.

[18] S.-T. Li. A platform-neutral live IP/TV presentation system. *Information Sciences*, 140(1-2):33 -- 52, 2002.

[19] Y. R. Liu and W. G .Tzeng. Public key broadcast encryption with low number of keys and constant decryption time. In *PKC'08*, volume 4939 of *LNCS*, pages 380--396. Springer-Verlag, 2008.

[20] J. Lotspiech, S. Nusser, and F. Pestoni. Broadcast encryption's bright future. *Computer*, 35:57--63, 2002.

[21] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO'01*, volume 2139 of *LNCS*, pages 41--62. Springer-Verlag, 2001.

[22] J. H. Park, H. J. Kim, M. H. Sung, and D. H. Lee. Public key broadcast encryption schemes with shorter transmissions. *IEEE Transactions on Broadcasting*, 54(3):401--411, 2008.

[23] Z. Ramzan and D. Woodruff. Fast algorithms for the free riders problem in broadcast encryption. In *CRYPTO'06*, volume 4117 of *LNCS*, pages 308--325. Springer-Verlag, 2006.

[24] C. B. S. Traw. Protecting digital content within the home. *Computer*, 34:42--47, 2001.

[25] D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures, 1999. Internet draft.

[26] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communication using key graphs. In *SIGCOMM'98*, pages 68--79, September 1998.